This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims

1. (Previously Presented) A method for accessing data in a database table, comprising:

receiving a fetch request to fetch data from a base table that satisfies a query predicate, wherein rows of the base table are stored in table partitions and wherein there is one index partition for each determined table partition, wherein each index partition includes nodes, wherein each node in each index partition includes at least one key column value from a corresponding table row in the table partition associated with the index partition and a location identifier identifying the corresponding table row in the corresponding table partition;

comparing a direction indicated in the fetch request and an ordering of the index partitions;

setting a fetch direction based on a result of the comparison of the direction indicated in the fetch request and the ordering of the index partitions;

scanning the index partitions in the fetch direction to determine a set of nodes from the index partitions whose key column value satisfies the query predicate;

ordering the set of determined nodes from the index partitions;

selecting one node from the ordered set based on a position of the node in the ordering; and

returning data from the table row identified by the location identifier in the selected node in response to the fetch request.

2. (Canceled)

3. (Previously Presented) The method of claim 1, wherein the fetch direction is set opposite the direction indicated in the fetch request if the direction indicated in the fetch request is opposite the ordering of the index partitions.

4. (Previously Presented) The method of claim 1, wherein setting the fetch direction comprises:

setting the fetch direction to backward if the fetch direction is backward and the fetch direction is not opposite the ordering of the index partitions or if the fetch direction is forward and the fetch direction is opposite the ordering of the index partitions; and

setting the fetch direction to forward if the fetch direction is backward and the fetch direction is opposite the ordering of the index partitions or if the fetch direction is forward and the fetch direction is not opposite the ordering of the index partitions.

5. (Previously Presented) The method of claim 1, further comprising:

if the fetch request is a first fetch of the fetch request, then selecting one node starting from one of: a lowest key value from each index partition if the fetch direction is forward or highest key value from each index partition if the fetch direction is backward.

6. (Previously Presented) The method of claim 1, further comprising:

if the fetch request is not a first fetch of the fetch request, then determining whether the fetch direction in which the index partitions are scanned for a previous fetch request is a same direction as the direction indicated in a current fetch request, wherein the direction indicated in the fetch request is capable of having been modified; and

if the fetch direction for the previous fetch request and direction indicated in the current fetch request are different, then discarding all saved nodes for the index partitions and selecting one node from a last selected node.

7. (Original) The method of claim 6, further comprising:

if the previous and current directions are the same, then scanning in the direction of the fetch request from the previously saved node in each index partition.

8. (Original) The method of claim 1, further comprising:

receiving a subsequent fetch request to fetch data from the base table;

replacing a previously selected node selected in a previous fetch request in the set with one node in the index partition including the previously selected node whose key column value satisfies the query predicate to form a modified set;

selecting one node from the modified set; and

returning the table row identified by the location identifier in the node selected from the modified set.

9. (Original) The method of claim 8, wherein the subsequent fetch request comprises a fetch relative request to fetch a row that is multiple number of rows from the previously selected node, further comprising:

performing the steps of replacing the previously selected node and selecting one node multiple number of times to determine the selected node to return to the fetch relative request to satisfy a fetch quantity.

10. (Original) The method of claim 8, wherein the subsequent fetch request comprises a fetch absolute request to fetch a row that is multiple number of rows from one end of the table, further comprising:

determining a new set of nodes, one from each index partition, by scanning from one end of the index partitions for a first node whose key column value satisfies the query predicate and whose key column value is greater than the previously selected node if fetching forward and the key is less than the previously selected node if fetching backward;

performing the steps of replacing the previously selected node and selecting one node a number of times that is one less than the number of rows indicated in the fetch absolute request to determine the selected node to return to the fetch relative request; and

performing the steps of replacing the previously selected node and selecting one node the multiple number of times to determine the selected node to return to the fetch relative request.

11. (Previously Presented) The method of claim 1, further comprising:

discarding cached keys if the fetch request is in an opposite direction of a previous fetch request;

determining a new set of nodes from each index partition; and

caching the determined new set of nodes when performing the fetch operation.

12. (Previously Presented) The method of claim 11, further comprising:

processing the fetch request to determine set of nodes in the backward direction in the previous fetch request;

inverting the keys and sorting the inverted keys; and

selecting the one node containing the lowest inverted key to return.

13-30. (Canceled)

31. (Previously Presented) The method of claim 1, further comprising:

determining whether the key value of the selected node from the ordered set satisfies the query predicate; and

selecting a next node form the ordered set following the selected node that does not satisfy the query predicate.

32. (Previously Presented) The method of claim 1, wherein determining the set of nodes from the index partitions comprises executing parallel tasks to process the index partitions.

33-36. (Canceled)

37. (Previously Presented) A system for accessing data in a database table, comprising:

a computer readable storage medium;

a base table implemented in the computer readable medium;

table partitions storing rows of the base table implemented in the computer readable medium;

index partitions, wherein there is one index partition for each determined table partition, wherein each index partition includes nodes, wherein each node in each index partition includes at least one key column value from a corresponding table row in the table partition associated

with the index partition and a location identifier identifying the corresponding table row in the corresponding table partition;

a database server for performing operations, the operations comprising:

receiving a fetch request to fetch data from a base table that satisfies a query predicate, wherein rows of the base table are stored in table partitions and wherein there is one index partition for each determined table partition, wherein each index partition includes nodes, wherein each node in each index partition includes at least one key column value from a corresponding table row in the table partition associated with the index partition and a location identifier identifying the corresponding table row in the corresponding table partition;

comparing a direction indicated in the fetch request and an ordering of the index partitions;

setting a fetch direction based on a result of the comparison of the direction indicated in the fetch request and the ordering of the index partitions;

scanning the index partitions in the fetch direction to determine a set of nodes from the index partitions whose key column value satisfies the query predicate;

ordering the set of determined nodes from the index partitions;

selecting one node from the ordered set based on a position of the node in the ordering; and

returning data from the table row identified by the location identifier in the selected node in response to the fetch request.

38. (Previously Presented) The system of claim 37, wherein the fetch direction is set opposite the direction indicated in the fetch request if the direction indicated in the fetch request is opposite the ordering of the index partitions.

39. (Previously Presented) The system of claim 37, wherein setting the fetch direction comprises:

setting the fetch direction to backward if the fetch direction is backward and the fetch direction is not opposite the ordering of the index partitions or if the fetch direction is forward and the fetch direction is opposite the ordering of the index partitions; and

setting the fetch direction to forward if the fetch direction is backward and the fetch direction is opposite the ordering of the index partitions or if the fetch direction is forward and the fetch direction is not opposite the ordering of the index partitions.

40. (Previously Presented) The system of claim 37, wherein the operations further comprise:

if the fetch request is a first fetch of the fetch request, then selecting one node starting from one of: a lowest key value from each index partition if the fetch direction is forward or highest key value from each index partition if the fetch direction is backward.

41. (Previously Presented) The system of claim 37, wherein the operations further comprise:

if the fetch request is not a first fetch of the fetch request, then determining whether the fetch direction in which the index partitions are scanned for a previous fetch request is a same direction as the direction indicated in a current fetch request, wherein the direction indicated in the fetch request is capable of having been modified; and

if the fetch direction for the previous fetch request and direction indicated in the current fetch request are different, then discarding all saved nodes for the index partitions and selecting one node from a last selected node.

42. (Previously Presented) The system of claim 41, wherein the operations further comprise:

if the previous and current directions are the same, then scanning in the direction of the fetch request from the previously saved node in each index partition.

43. (Previously Presented) The system of claim 37, wherein the operations further comprise:

receiving a subsequent fetch request to fetch data from the base table;

replacing a previously selected node selected in a previous fetch request in the set with one node in the index partition including the previously selected node whose key column value satisfies the query predicate to form a modified set;

selecting one node from the modified set; and

returning the table row identified by the location identifier in the node selected from the modified set.

44. (Previously Presented) The system of claim 43, wherein the subsequent fetch request comprises a fetch relative request to fetch a row that is multiple number of rows from the previously selected node, further comprising:

performing the steps of replacing the previously selected node and selecting one node multiple number of times to determine the selected node to return to the fetch relative request to satisfy a fetch quantity.

45. (Previously Presented) The system of claim 43, wherein the subsequent fetch request comprises a fetch absolute request to fetch a row that is multiple number of rows from one end of the table, wherein the operations further comprise:

determining a new set of nodes, one from each index partition, by scanning from one end of the index partitions for a first node whose key column value satisfies the query predicate and whose key column value is greater than the previously selected node if fetching forward and the key is less than the previously selected node if fetching backward;

performing the steps of replacing the previously selected node and selecting one node a number of times that is one less than the number of rows indicated in the fetch absolute request to determine the selected node to return to the fetch relative request; and

performing the steps of replacing the previously selected node and selecting one node the multiple number of times to determine the selected node to return to the fetch relative request.

46. (Previously Presented) The system of claim 37, wherein the operations further comprise:

discarding cached keys if the fetch request is in an opposite direction of a previous fetch request;

determining a new set of nodes from each index partition; and

caching the determined new set of nodes when performing the fetch operation.

47. (Previously Presented) The system of claim 46, wherein the operations further comprise:

processing the fetch request to determine set of nodes in the backward direction in the previous fetch request;

inverting the keys and sorting the inverted keys; and

selecting the one node containing the lowest inverted key to return.

48. (Previously Presented) An article of manufacture comprising a computer readable storage medium having code executed by a processor for accessing data in a database table and for performing operations, the operations comprising:

receiving a fetch request to fetch data from a base table that satisfies a query predicate, wherein rows of the base table are stored in table partitions and wherein there is one index partition for each determined table partition, wherein each index partition includes nodes, wherein each node in each index partition includes at least one key column value from a corresponding table row in the table partition associated with the index partition and a location identifier identifying the corresponding table row in the corresponding table partition;

comparing a direction indicated in the fetch request and an ordering of the index partitions;

setting a fetch direction based on a result of the comparison of the direction indicated in the fetch request and the ordering of the index partitions;

scanning the index partitions in the fetch direction to determine a set of nodes from the index partitions whose key column value satisfies the query predicate;

ordering the set of determined nodes from the index partitions;

selecting one node from the ordered set based on a position of the node in the ordering; and

returning data from the table row identified by the location identifier in the selected node in response to the fetch request.

49. (Previously Presented) The article of manufacture of claim 48, wherein the fetch direction is set opposite the direction indicated in the fetch request if the direction indicated in the fetch request is opposite the ordering of the index partitions.

50. (Previously Presented) The article of manufacture of claim 48, wherein setting the fetch direction comprises:

setting the fetch direction to backward if the fetch direction is backward and the fetch direction is not opposite the ordering of the index partitions or if the fetch direction is forward and the fetch direction is opposite the ordering of the index partitions; and

setting the fetch direction to forward if the fetch direction is backward and the fetch direction is opposite the ordering of the index partitions or if the fetch direction is forward and the fetch direction is not opposite the ordering of the index partitions.

51. (Previously Presented) The article of manufacture of claim 48, wherein the operations further comprise:

if the fetch request is a first fetch of the fetch request, then selecting one node starting from one of: a lowest key value from each index partition if the fetch direction is forward or highest key value from each index partition if the fetch direction is backward.

52. (Previously Presented) The article of manufacture of claim 48, wherein the operations further comprise:

if the fetch request is not a first fetch of the fetch request, then determining whether the fetch direction in which the index partitions are scanned for a previous fetch request is a same direction as the direction indicated in a current fetch request, wherein the direction indicated in the fetch request is capable of having been modified; and

if the fetch direction for the previous fetch request and direction indicated in the current fetch request are different, then discarding all saved nodes for the index partitions and selecting one node from a last selected node.

53. (Previously Presented) The article of manufacture of claim 52, wherein the operations further comprise:

if the previous and current directions are the same, then scanning in the direction of the fetch request from the previously saved node in each index partition.

54. (Previously Presented) The article of manufacture of claim 48, wherein the operations further comprise:

receiving a subsequent fetch request to fetch data from the base table;

replacing a previously selected node selected in a previous fetch request in the set with one node in the index partition including the previously selected node whose key column value satisfies the query predicate to form a modified set;

selecting one node from the modified set; and

returning the table row identified by the location identifier in the node selected from the modified set.

55. (Previously Presented) The article of manufacture of claim 54, wherein the subsequent fetch request comprises a fetch relative request to fetch a row that is multiple number of rows from the previously selected node, wherein the operations further comprise:

performing the steps of replacing the previously selected node and selecting one node multiple number of times to determine the selected node to return to the fetch relative request to satisfy a fetch quantity.

56. (Previously Presented) The article of manufacture of claim 54, wherein the subsequent fetch request comprises a fetch absolute request to fetch a row that is multiple number of rows from one end of the table, wherein the operations further comprise:

determining a new set of nodes, one from each index partition, by scanning from one end of the index partitions for a first node whose key column value satisfies the query predicate and whose key column value is greater than the previously selected node if fetching forward and the key is less than the previously selected node if fetching backward;

performing the steps of replacing the previously selected node and selecting one node a number of times that is one less than the number of rows indicated in the fetch absolute request to determine the selected node to return to the fetch relative request; and

performing the steps of replacing the previously selected node and selecting one node the multiple number of times to determine the selected node to return to the fetch relative request.

57. (Previously Presented) The article of manufacture of claim 48, wherein the operations further comprise::

discarding cached keys if the fetch request is in an opposite direction of a previous fetch request;

determining a new set of nodes from each index partition; and

caching the determined new set of nodes when performing the fetch operation.

58. (Previously Presented) The article of manufacture of claim 57, wherein the operations further comprise:

processing the fetch request to determine set of nodes in the backward direction in the previous fetch request;

inverting the keys and sorting the inverted keys; and

selecting the one node containing the lowest inverted key to return.

59. (Previously Presented) The article of manufacture of claim 48, wherein the operations further comprise:

determining whether the key value of the selected node from the ordered set satisfies the query predicate; and

selecting a next node form the ordered set following the selected node that does not satisfy the query predicate.